

**Московский государственный технический
университет им Н.Э. Баумана**

Факультет «Информатика и системы управления»
Кафедра ИУ5 «Системы обработки информации и управления»

Курс «Базовые компоненты интернет-технологий»

Лабораторная работа №4
«Шаблоны проектирования и модульное тестирование в Python»

Выполнил:

студент группы ИУ5-34Б
Данилин Максим

Подпись и дата:

Проверил:
Гапанюк Ю. Е.

Подпись и дата:

Москва, 2021 г.

Постановка задачи

1. Необходимо для произвольной предметной области реализовать от одного до трех шаблонов проектирования: один порождающий, один структурный и один поведенческий. В качестве справочника шаблонов можно использовать следующий каталог. Для сдачи лабораторной работы в минимальном варианте достаточно реализовать один паттерн.
2. Вместо реализации паттерна Вы можете написать тесты для своей программы решения биквадратного уравнения. В этом случае, возможно, Вам потребуется доработать программу решения биквадратного уравнения, чтобы она была пригодна для модульного тестирования.
3. В модульных тестах необходимо применить следующие технологии:
 - TDD - фреймворк.
 - BDD - фреймворк.
 - Создание Mock-объектов.

Текст программы

Файл bq.py:

```
import sys
import math

def get_coef(index, prompt):
    try:
        coef = float(sys.argv[index])
    except:
        while True:
            try:
                coef = float(input(prompt))
                break
            except ValueError:
                print("Ошибка! Попробуйте ещё раз...")
    return coef

def check(result, root):
    if root == 0.0:
        result.append(root)
    elif root > 0.0:
        result.append(-round(math.sqrt(root), 3))
        result.append(round(math.sqrt(root), 3))

def get_roots(a, b, c):
    result = []
    if a == 0:
        if b != 0:
            root = -c / b
            check(result, root)
        return result
    D = b * b - 4 * a * c
    if D == 0.0:
        root = -b / (2.0 * a)
        check(result, root)
    elif D > 0.0:
        sqrtD = math.sqrt(D)
        root1 = (-b + sqrtD) / (2.0 * a)
        root2 = (-b - sqrtD) / (2.0 * a)
        check(result, root1)
        check(result, root2)
    return result

def main():
    a = get_coef(1, 'Введите коэффициент A:')
    b = get_coef(2, 'Введите коэффициент B:')
    c = get_coef(3, 'Введите коэффициент C:')

    if a == b == c == 0.0:
        print('Бесконечное число корней')
    else:
        roots = get_roots(a, b, c)
        len_roots = len(roots)
        if len_roots == 0:
            print('Нет корней')
```

```

        elif len_roots == 1:
            print('Один корень: {}'.format(roots[0]))
        elif len_roots == 2:
            print('Два корня: {} и {}'.format(roots[0], roots[1]))
        elif len_roots == 3:
            print('Три корня: {}, {} и {}'.format(roots[0], roots[1],
roots[2]))
        elif len_roots == 4:
            print('Четыре корня: {}, {}, {} и {}'.format(roots[0], roots[1],
roots[2], roots[3]))


if __name__ == "__main__":
    main()

```

Файл test_bq.py:

```

import unittest
from bq import get_roots, check
import math

class TestRoots(unittest.TestCase):

    def test_roots_is_equal(self):
        self.assertEqual(get_roots(1,-5,6), [-1.732, 1.732, -1.414, 1.414])
        self.assertEqual(get_roots(1,-4,4), [-1.414, 1.414])
        self.assertEqual(get_roots(-4,16,0), [0.0, -2.0, 2.0])
        self.assertEqual(get_roots(1,0,-16), [-2.0, 2.0])

    def test_string_root(self):
        self.assertRaises(TypeError, get_roots, 'fafsdlkfj')
        self.assertRaises(TypeError, get_roots, 'True')
        self.assertRaises(TypeError, get_roots, [1,2])

if __name__ == '__main__':
    unittest.main()

```

Результат выполнения

```

Launching unitests with arguments python -m unittest /Users/choza_max/PycharmProjects/lab4/test_bq.py in
/Users/choza_max/PycharmProjects/lab4

```

```
Ran 2 tests in 0.001s
```

```
OK
```

Файл bdd_test_bq.py:

```

from behave import given, when, then
from bq import get_roots, check

@given(u"Biquadratic equation app is run")

```

```

def step_impl(context):
    print(u'Step: Given Biquadratic equation app is run')

@when(u'I have the odds "{a}", "{b}", and "{c}"')
def step_impl(context, a, b, c):
    print(u'Step: I have the odds "{}", "{}", and "{}". format(a, b, c)')
    b = str(get_roots(int(a), int(b), int(c))).rpartition(']')[0]
    c = b.partition('[')[2]
    context.result = c
    print(u'Stored result "{}" in context'.format(context.result))

@then(u'I get result "{out}"')
def step_impl(context, out):
    if (context.result == str(out)):
        print(u'Step: Then I get right result "{}",
"{}".format(context.result, out))'
        pass
    else :
        raise Exception ("Invalid root is returned.")

```

Файл bdd_test_bq.feature:

```

Feature: Test Biquadratic equation Functionality

Scenario: Test my biquadratic equation
    Given Biquadratic equation app is run
    When I have the odds "1", "-5", and "6"
    Then I get result "-1.732, 1.732, -1.414, 1.414"

Scenario: Test my biquadratic equation
    Given Biquadratic equation app is run
    When I have the odds "1", "-4", and "4"
    Then I get result "-1.414, 1.414"

Scenario: Test my biquadratic equation
    Given Biquadratic equation app is run
    When I have the odds "1", "0", and "-16"
    Then I get result "-2.0, 2.0"

```

Результат выполнения

```
(venv) choza_max@MacBook-Pro-Maksim lab4 % behave bdd_test_bq.feature
Feature: Test Biquadratic equation Functionality # bdd_test_bq.feature:1

  Scenario: Test my biquadratic equation          # bdd_test_bq.feature:3
    Given Biquadratic equation app is run        # steps/bdd_test_bq.py:5 0.000s
    When I have the odds "1", "-5", and "6"      # steps/bdd_test_bq.py:9 0.000s
    Then I get result "-1.732, 1.732, -1.414, 1.414" # steps/bdd_test_bq.py:17 0.000s

  Scenario: Test my biquadratic equation          # bdd_test_bq.feature:8
    Given Biquadratic equation app is run        # steps/bdd_test_bq.py:5 0.000s
    When I have the odds "1", "-4", and "4"       # steps/bdd_test_bq.py:9 0.000s
    Then I get result "-1.414, 1.414"            # steps/bdd_test_bq.py:17 0.000s

  Scenario: Test my biquadratic equation          # bdd_test_bq.feature:13
    Given Biquadratic equation app is run         # steps/bdd_test_bq.py:5 0.000s
    When I have the odds "1", "0", and "-16"      # steps/bdd_test_bq.py:9 0.000s
    Then I get result "-2.0, 2.0"                 # steps/bdd_test_bq.py:17 0.000s

1 feature passed, 0 failed, 0 skipped
3 scenarios passed, 0 failed, 0 skipped
9 steps passed, 0 failed, 0 skipped, 0 undefined
Took 0m0.001s
```